Familienname:

Vorname:

Matrikelnummer:

Studienkennzahl:

Aufgabe 1 (4 Punkte):
Aufgabe 2 (3 Punkte):
Aufgabe 3 (2 Punkte):
Aufgabe 4 (2 Punkte):
Aufgabe 5 (3 Punkte):
Aufgabe 6 (1 Punkt):
Aufgabe 7 (1 Punkt):
Aufgabe 8 (4 Punkte):
Aufgabe 9 (10 Punkte):

Gesamtpunktzahl:

Note:

Schriftlicher Test 1 (90 Minuten)
VU Einführung ins Programmieren für TM

24. November 2006

Aufgabe 1 (4 Punkte). Was sind die Bestandteile einer Gleitkommazahl des Zahlsystems $\mathbb{F}(b, p, e_{\min}, e_{\max})$ und wie ermittelt man aus diesen Bestandteilen den Wert der Gleitkommazahl?

Aufgabe 2 (3 Punkte). Was sind normalisierte und denormalisierte Gleitkommazahlen? Was ist ein implizites erstes Bit? Kann man bei Gleitkommazahlsystemen mit Basis $b = 10$ ein implizites erstes Bit verwenden? (Begründung!)

Aufgabe 3 (2 Punkte). Was ist die größte Gleitkommazahl im IEEE-Gleitkommazahlsystem $\mathbb{F} = \mathbb{F}(2, 53, -1021, 1024)$? Was ist die kleinste positive denormalisierte Gleitkommazahl in $\mathbb{F}$?

Aufgabenstellung. Eine Matrix $U \in \mathbb{R}^{n \times n}$, deren Einträge unterhalb der Hauptdiagonale gleich 0 sind, bezeichnet man als obere Dreiecksmatrix,

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ & u_{22} & u_{23} & \dots & u_{2n} \\ & & u_{33} & \dots & u_{3n} \\ & & & \ddots & \vdots \\ \mathbf{0} & & & & u_{nn} \end{pmatrix}$$

bzw. mathematisch formuliert: $u_{jk} = 0$ für $j > k$. In Aufgabe 4 soll eine Datenstruktur zur dynamischen Speicherung von U hergeleitet werden, die in allen weiteren Aufgaben verwendet werden soll. In Aufgabe 5–8 sollen einige der Zugriffsfunktionen programmiert werden. In Aufgabe 9 soll ein Löser für das lineare Gleichungssystem $Ux = b$ hergeleitet und geschrieben werden.

Erinnerung. Die Matrix-Vektor-Multiplikation $b := Ux \in \mathbb{R}^n$ einer Matrix $U \in \mathbb{R}^{n \times n}$ mit einem Vektor $x \in \mathbb{R}^n$ ist definiert durch

$$b_j := \sum_{k=1}^{n} u_{jk} x_k \quad \text{für } j = 1, \dots, n.$$

Aufgabe 4 (2 Punkte). Definieren Sie einen Struktur-Datentyp `matrixU` zur Speicherung von oberen Dreiecksmatrizen $U \in \mathbb{R}^{n \times n}$. Dabei sollen die `double`-Einträge der Matrix in einem (dynamischen) Array der Länge $\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$ gespeichert werden.

Aufgabe 5 (3 Punkte). Schreiben Sie eine Funktion `newMatrixU`, die eine obere Dreiecksmatrix allokiert und initialisiert.

Aufgabe 6 (1 Punkt). Schreiben Sie eine Funktion `delMatrixU`, die den Speicher einer oberen Dreiecksmatrix freigibt.

Aufgabe 7 (1 Punkt). Schreiben Sie eine Funktion `getMatrixDimension`, die die Dimension n der Matrix $U \in \mathbb{R}^{n \times n}$ zurückliefert.

Aufgabe 8 (4 Punkte). Schreiben Sie eine Funktion `getMatrixUjk`, die für *alle* Indizes $j, k = 1, \ldots, n$ einen Eintrag U_{jk} einer oberen Dreiecksmatrix zurückliefert. Sie müssen nicht überprüfen ob $j, k \in \{1, \ldots, n\}$ gilt.

Aufgabe 9 (10 Punkte). Gegeben sei eine obere Dreiecksmatrix $U \in \mathbb{R}^{n \times n}$ mit $u_{jj} \neq 0$ für alle $j = 1, \ldots, n$. Zu gegebenem $b \in \mathbb{R}^n$ existiert dann ein eindeutiges $x \in \mathbb{R}^n$ mit $Ux = b$. Schreiben Sie eine Funktion `solveU`, die für einen gegebenen `double`-Vektor $b \in \mathbb{R}^n$ das Gleichungssystem $Ux = b$ löst. Der Vektor sei in einer Struktur `vector` gespeichert, deren Zugriffsfunktionen die folgenden Signaturen haben:

```
vector* newVector(int n);
vector* delVector(vector* x);
int getVectorLength(vector* x);
double getVectorXj(vector* x, int j);
void setVectorXj(vector* x, int j, double Aij);
```

Sie dürfen bei Ihrer Implementierung davon ausgehen, dass der Vektor x und die Matrix U übereinstimmende Dimensionen haben. Auf die Nulleinträge von U soll nicht zugegriffen werden, um den Rechenaufwand gering zu halten. — Um den Algorithmus herzuleiten, schreibe man das Matrix-Vektor-Produkt $b = Ux$ komponentenweise für b_j mit $j = 1, \ldots, n$ als Summe hin. Man überlege, wie die spezielle Gestalt von U die Laufindizes der Summe vereinfacht und löse diese Gleichung nach x_j auf.

Verwenden Sie die linke Seite für die Herleitung des Algorithmus und die rechte Seite für die Formulierung der Funktion.